NASA/KSC Cooperative Agreement

MEMORY-BASED REASONING FOR ADVANCED LAUNCH SYSTEM OPERATIONS

1990 Annual Report

Dr. Harley R. Myler, Principal Investigator
Department of Computer Engineering
University of Central Florida
(407) 823-5098
hrm@engr.ucf.edu

Dean A. Dubois, Research Assistant
Department of Computer Engineering
University of Central Florida
dad@engr.ucf.edu

January 25, 1991

I. Introduction

This project addresses a need for investigation into advanced techniques of automated launch processing. Our thesis has been that Artificial Neural Networks (ANNs) are a prime candidate for the architecture of a launch processing system that would use a Memory-Based Reasoning paradigm (Stanfill & Waltz, 1986) in order to observe, evaluate and suggest corrective action during a launch sequence. It should be clear that the sequence of events that lead to the successful launch of a space vehicle or other payload is a complex, time-dependent process. In addition, it is a process that is highly dependent on human capability and attention. It is important to note that we seek to produce an automated launch paradigm that would relieve a portion of the human attentional burden from the launch personnel.

The long-term objectives of this project have been to:

- Create a model scenario of Launch Operations subsystems.
- Generate an I/O structure for the model.
- Generate an information exchange protocol for the I/O.
- Build an appropriate neural model for reasoning on the activities of the model scenario.
- Test the model.
- Extend the model to large parallel processing hardwares.

A necessary objective of this research is to develop a general paradigm of automated reasoning based on memory associations. At present, we have accomplished the first four objectives and are working on a model system to run on a parallel computer purchased for this project. The details of our model and the parallel processing system are discussed in the body of this report.

II. Memory-Based Reasoning for Advanced Launch Operations

Memory-Based Reasoning (MBR) is reasoning that takes place as a function of data retrieval from a memory system. It has been discussed at length in previous reports published for this project and at presentations. Additionally, a paper entitled "A Neural Model for Memory-Based Reasoning in Advanced Launch Operations" is currently at press (Myler, 1991) that discusses and illustrates some basic and early results of this research. The concept of using MBR as a mechanism for automated launch monitoring was initially suggested in the proposal submitted to fund this research. In addition, the use of an artificial neural network (ANN) as the associative store for the system is also a unique invention of this research. Two problems arise when working with both MBR paradigms and ANNs--the problems of data formatting and data interpretation. In addition to these problems, we have

uncovered an additional impediment to the development of a system in the problem of representing and evaluating temporal sequencing.

In the case of data formatting, the immediate question is, "What form should the data take as an input to the neural system such that a correct interpretation of the output resulting from that data can be made?". Initially, our proposition was that the MBR system would work as a high level executive and intelligently process and analyze data from sophisticated sources. These sources would include automated subsystems imbued with their own reasoning capabilities, yet restricted to specific domains. For example, weather processing, scheduling, logistics operations, and process systems diagnosis and control. Automated reasoning systems have been researched by NASA to address all four of these areas. An early version of this project, the Advanced Launch Operations Intelligent System (ALOIS), was modelled using a hypermedia development tool and illustrated how intelligent subsystems might interact and communicate with each other.

Originally, the problem of data formatting was to be solved by use of hypermedia as a data constraint mechanism. The four systems to be intelligently integrated would interact with the MBR system via high-level indicators and responses to commands. In our demonstration, this interaction was accomplished via on-screen, mouse selectable buttons and text fields. Liberal use of animated graphics and digitized and synthesized sound added to the effect of a well choreographed interaction of subsystems. Our eventual intent was to replace the fixed decision scripts (the hypermedia program) with a neural system that would react using memory associations of event order. If event order or occurrence was inconsistent with the existing memory contents, the system would react with an anomaly indication. The overriding approach was to be a system that would integrate a large database of knowledge (of launch system processes) and compare it to an existing state of the monitored system. The system then react by drawing attention to an anomaly response developed from a comparison of the current state to a state that it had previously observed. In short, the system would learn from analogy and example. The setup conditions for learning to take place is determined from how the data interface of the hypermedia system has been formulated.

The hypermedia approach to MBR is similar to that proposed for the *Knoesphere* system (Lenat, *et al*, 1983), a proposed hypermedia-based expert system with encyclopedic knowledge. The *Knoesphere* is intended to be frame-based, with heuristic rules that govern the building and checking of user models (the system is intended to adapt itself to a user's ability level) and the decision of which modalities to activate based on the data presented. The difference between the MBR system and the *Knoesphere* is that *a priori* knowledge is only supplied as a function of the hypermedia interface structure. The actual knowledge data is completely unstructured and exists only as weights within the neural sponge of the

ANN used for the associative store. Our conjecture is that the system must learn it's knowledge in the way that a human does and that this knowledge cannot be injected explicitly. A great deal of discussion has addressed this statement (Dreyfus, 1979; Dreyfus & Dreyfus, 1988; Minsky, 1985) and we will not belabor it here. Of greater importance is the validity of our conclusion that a neural system is capable of both learning and associating knowledge intelligently.

We draw some measure of confidence that machine learning can take place from studies in neurobiology. Learning and conditioning are thought to be encoded in the neural subsystems of all animals possessing nerve cells (Kanigel, 1987). We must ask the question, "How does a single neuron, an elementary computing element of extreme simplicity, contribute to the incredible knowledge processing power that a brain possesses?". The answer to this is not in the individual neuron, but rather in the collective behavior of huge numbers of neurons and their interconnectivity. Of importance is our understanding of the complexity and immensity of the human brain, our only model of advanced intelligence. To underscore this, we quote:

> One peculiarity of biological neural networks is their size: in the whole human central nervous system there are on the order of 10^{11} neurons, but the number of their interconnections is still higher, probably up to the order of 10^{15}. It does not seem posssible to *program* the function of such a system according to a prior plan, not even by genetic information; consider, too, that the size and structure of the network is changing radically during and after childhood, when it is already in use (Kohonen, 1988).

If one is to program a neural system, then this programming must take the form of a change to the interconnectivity structure of the network either in terms of the mapping of interconnects or the signal weights between the neurons. Nature has had the advantage of billions of years of evolution to accomplish the complex structural programming of animal neural systems. In order to approach a limited emulation of this programming, we must content ourselves to focussing on the external data formatting of the information to be stored and associated.

To summarize, an MBR system associatively stores information and exhibits a reasoning behavior by making inferences from previous experiences of data. The simplest example of this is the following scenario:

assertion: The scheduled **launch** is in **three hours.**
assertion: The weather forecast indicates **bad weather** in **three hours.**
conclusion: **Launch cannot take place** with **bad weather.**

It is easy to see how a rule that operates on comparing the predicted weather conditions with the time of anticipated launch could easily reach the correct

called) is capable of modelling virtually any type of neural network to virtually any size depending on the capabilities of the computer host. The first topology proposed for MBR was a three layer fully connected network, similar to that shown in Figure 1. The input layer data consisted of four fields of five characters going to individual nodes. The middle, or hidden, layer had five nodes and the output layer had an identical structure to the input layer. Groups of characters were chosen to represent various conditions that a space shuttle awaiting launch might encounter. These data elements were determined in an *ad hoc* fashion from our, albeit limited, knowledge of launch operations. It must be emphasized that our goal was to examine the functionality of the model in a launch processing context and not to simulate an actual launch scenario.

Abbreviations were selected to indicate the status of WEATHER, SCHEDULING, and SYSTEMS and insure that no ambiguities existed between data elements. Each of these inputs is meant to simulate the input from automated reasoning sub-systems that evaluate conditions within their realm of understanding and report to the launch director. The possible conditions that can be reported by each of the input fields is:

Weather:
1) CLEAR - clear skies (good visibility is required for a launch)
2) THSTO - thunderstorm (may require mission to be scrubbed)
3) FREZE - freezing conditions (restrictions to operational temp)
4) RASTO - rainstorm (fowl weather forces a delay in the count)
5) CLODY - cloudy (visibility in case an emergency landing must be made)
6) FOGGY - foggy (altitude of cover effects documentary filming)

Scheduling:
1) NODEL - no delay (all systems ready)
2) SHDEL - short delay (required for system readiness)
3) LNDEL - long delay (major time delay evident)

Systems:
1) GOODY - all systems functioning
2) BDRED - bad reading (intermittent or faulting input)
3) BDELE - bad element (diagnostic failure)

The fourth input field indicates the launch status:

1) NOHLD - no hold (count proceeds uninterrupted)
2) IDHLD - indefinite hold (mission countdown suspended)
3) SCRUB - scrub (mission canceled)

Once the topology and data structure had been established, various forms of back propagation learning algorithms (see Carpenter, *et al*, 1987) were used to train the network. Training of multiple input/output pairs allowed all of the patterns that had been developed to be presented for training automatically. During this automated training mode, the maximum number of iterations to search for a converged network as well as the maximum error tolerance were entered. These

two parameters (along with the complexity of the network itself) determined the amount of time it would take to train the network. The results of multiple training runs of the initial topology are tabulated in Appendix A. This data shows an exponential relationship between the number of patterns that a network is expected to learn and the number of passes that are required to train those patterns. While the time to train the network may not be a priority issue at this time, the turn-around time for testing a wide variety of network configurations requires that the time for a network to successfully converge be kept at a minimum.

In an attempt to reduce the complexity of the network, the topology was changed to fifteen inputs (three fields of five characters each), a middle layer of five nodes and an output layer of five nodes. This topology allows only conditions of WEATHER, SCHEDULING and SYSTEMS to be input to the neural network, with no fourth data field for launch status. The output field is concerned only with hold/no hold status decisions. Under this scheme, the training time for the network was drastically reduced and the number of patterns that could be successfully converged upon was more than doubled.

Once the initial test patterns had been trained, a search for signs of emergent cognitive activity was pursued. New input patterns were presented and the computed outputs were observed. The new input patterns consisted of slight spelling alterations of the original patterns as well as new combinations of the input fields. The slight errors in spelling often resulted in an output pattern that was exact or very close to the original training pair. New combinations, however, yielded partial string responses with some of the characters differing from the trained pairs by as much as three hundred percent. These unexpected differences indicate that the generalizations formed by the network involve strong inter-field relationships.

In order to verify this speculation, as well as develop new techniques for training complex networks, several small subnetworks were designed. These subnetworks were to be trained to provide logical internal representations and then be combined together to form a single network. In order to test subnetwork performance without creating a network topology specifically designed to solve this problem, the initial format of the inputs was retained. Changes in the structure of the overall network were made by introducing small subsystems into each data field of the input stream. In short, a neural network was created for each input field. Each subnetwork was then trained to read the ASCII characters that could be entered in its respective input field and associate that character group to a set of numeric parameters. In the case of SCHEDULING, there was only one parameter, time. For the SYSTEMS input field, both a diagnostic parameter and a faulty reading parameter were created. The WEATHER subsystem required the greatest diversity resulting in eight parameters. The eight datums for weather were derived

from local climatological observations. Four fundamental elements were included for each possible weather condition. Visibility, temperature, humidity and cloud cover were encoded as two values each. A parameter for the mean value and one for the deviation of each of the elements for the four weather conditions were computed. The subnetworks were created and trained individually and training results for each subnetwork as well as the climatological data are presented in Appendix B. The resulting networks may be thought of as a dictionary, a network that has not been specifically trained with any knowledge of the over-all problem to be solved. Instead, these networks are able to read input characters and relate them to unique parameters. This conversion from lexigraphical data to numerical approximations should allow the system to read and write the character strings it has been trained, as well as form associations of known parameters. These associations should be much easier to identify and verify within the network's operation than the unknown relationships between the arbitrary characters of the input and output fields.

The three subnets were combined to form a single network and initial weights were set according to the results of the individual trainings. Because this network consisted of three discrete portions (to take into account the three distinct input fields), the new network topology, Figure 2, does not appear to be fully connected between successive layers. The connections still exist, however, their weight has been forced to zero. As long as the learning factor for this layer is kept at zero, there will be no interaction between the characters of these input fields. The original input patterns were presented with the learning factor for the first layer (the subnets or dictionary layer) set to 0. This allows the training previously recorded to remain while inferences are drawn from the new parameter space. The accumulated training time is much less for systems that are made up of subnets. Since the subnets are generic in nature, they could be used in multiple network designs without being retrained. The possibility of libraries of subnets with understanding in a wide variety of areas would allow the creation of complex networks by combining high-level components into new and unique configurations. The subnet approach also gives rise to easy implementation in a parallel machine.

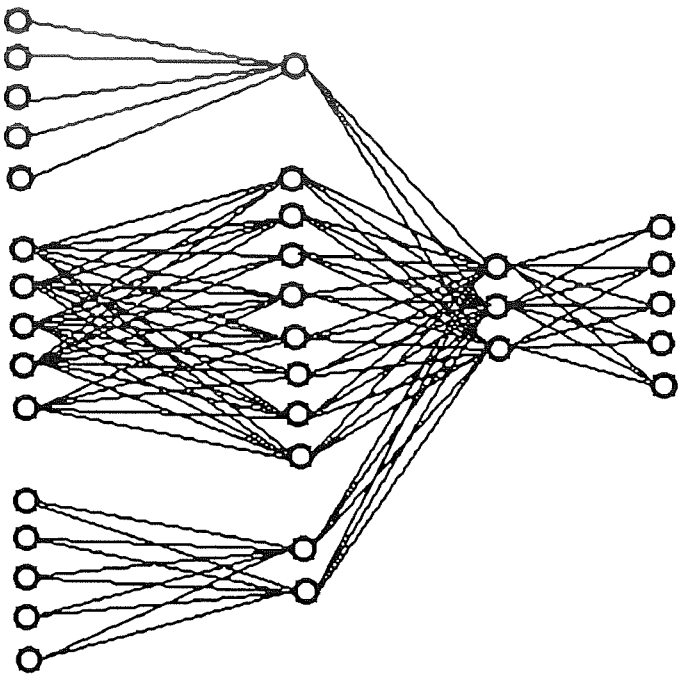

Figure 2. Subnet Network Topology

While this approach does not provide direct proof that an abstract correlation will be found by a neural network, it will allow us to apply known generalizations and measure quantitatively the success of the neural network in finding that relationship. Once this *level of learning* has been established, it should be possible to compute the likelihood that the system will find unforeseen correlations between input data. The success of this approach will also lead the way to modular construction of complex neural networks. Components may be separated and trained and then assembled into larger systems with all of the knowledge of the combined subsections without the high cost of training an entire blank system with equal capabilities.

Upon successful completion of this preliminary implementation of neural network launch system modeling, a more complex approach was developed. This new system took the results of earlier networks and tested the expandability of the concept. Expansion took two major forms, first, the number of nodes in the system was increased. This meant a much larger number of inputs to the system had to be tested. The second form of expansion required that the network respond within the context of a launch timing model and be temporally guided.

The expansion of the number of inputs to the system forced the creation of tools to specify, develop and edit large amounts of input data. It was determined that this data would be collated in the form of a DATA-BUFFER, and that an additional file (the DATA DICTIONARY) would be accessed in order to fully

describe the significance of the various portions of the data buffer. Finally, two programs had to be created to interpret the data buffer. The first program allowed the buffer to be modified both in raw hexadecimal form and in a high level manner. The second program converted the DATA-BUFFER into a file that could be presented to the neural network.

The complexity of incorporating the concept of time into the network presented the greatest challenge to this phase of our study. In order for the system to make intelligent decisions about the viability of a given launch scenario, data must be available to indicate the current state of the system, as well as the scheduling constraints. In essence, the launch model must maintain two timing criteria: a *real-time* schedule (which dictates launch window and weather prediction), as well as *T-time* (which represents the relative phase of the launch process). Each of the timing inputs to the network were translated into discrete time intervals and these representations are shown in Table 1. The network must be able to combine the information about launch window, weather conditions and launch status to form a complete picture of the possibility of launch. It should be noted that the time divisions for weather and launch phase are arbitrary boundaries while the launch window is specified in uniform increments. The ability of the network to combine these two forms of time representation should further prove the flexibility of the neural network system.

The *launch window* time is specified using 100 hours of yes/no inputs. This allows a flexible description of multiple windows of varying widths. The *weather prediction* is broken into four categories:

long term	40 hours or more	poor
mid term	6-39 hours	fair
short term	5 hours or less	good
current readings	NOW	excellent

The "T-" time is divided into six segments that span T-100 (shuttle preparation) to T+10 (shuttle clear of atmosphere):

T-100	T-40	T-2	T-10m	T-10s	T+10m
prep/load	roll out	board	fuel	ready	launch

Table 1. Network Timing Inputs

Once the timing had been incorporated into the network scheme, new forms of data had to be devised and described. The new data was presented from three categories: Weather, Systems and Command. The weather system represents the output of an automated reasoning system that is providing weather prediction and current conditions. The weather status is divided into four fields and each of these fields indicates a potential problem to a launch. The fields selected are: visibility,

wind velocity, lightning and temperature. Long, mid and short term predictions are provided as well as the current conditions for each field.

The Systems data consists of various valves and switches throughout the shuttle and support systems and indicate internal monitoring . Provision for bad readings has been made so that a failure may be simulated in any component or group of components. Additionally, redundant components may be bypassed. The decision to bypass or replace a component comes from the Command section. This final portion of the data represents the top of the launch decision ladder. Here, the next phase of the launch is determined and the system may either continue from one phase to the next (a normal launch), repeat the current phase (a hold), jump back to a previous phase (an extended hold) or scrub the launch. These inputs constitute the training data for the neural network. The conjecture here is that the system will learn to conduct an optimal sequence toward launch.

The project has reached a point where it has become necessary to expand the scope of the neural networks function. To this point, all input have produced an immediate response. Although this will allow the system to attack a large number of problems, it eliminates any concept of time-dependent functions. The next phase of the study will require that the system be able to relate stimuli and the state of the system to a time element. With the exception of one publication (Grossberg and Schmajuk,1989), this new category of problem has, to our knowledge, been virtually unexplored in the neural network field. This problem most closely resembles circuits with memory vs. circuits without memory in that a memoryless circuit need maintain (nor has the ability to do so) no knowledge of past events.

The following proposition must be addressed: Is the main distinction between a system that realizes time-dependent action and a direct stimulus/response system the existence of memory? On the broader scale, time is merely another dimension to any equation. Given the fact that a neural network is capable of learning a sufficiently complex equation, it follows that such a system should be able to create time-dependent functions. In this form, the concept of time would be introduced to the system as a single input and this value would effect all other inputs. If on the other hand, one were to introduce the concept of short-term-memory to a system, time dependent functions could also be resolved and the system would more closely resemble the human mental model. Under this scheme, time would be represented as the memory of the previous state of the machine and the combination of previous state and current state would create a stimulus for action.

Short-term memory is recognized as an important element in the human mental process(see Shiffren & Atkinson, 1969; Waldrop, 1985). Various forms of the short term memory paradigm are used in both the learning process and the

actual gathering of information for the brain. For instance, the response of the eye to light as well as the ear to sound are short-term integrating functions. The intensity of a given input is perceived as a function of both the actual pressure of the input and the duration of that pressure. These short term memory effects serve to synchronize the outside world with the relatively slow operating frequency of the brain. During the cognitive process, the brain keeps differing levels of knowledge. These different areas represent new versus old information that has been correlated into a set of rules. The different areas of knowledge have different rules for being updated and influence each other in the perception of the world. This separation of new and old knowledge most likely exists in many layers rather than the two distinct categories represented here. The thought process is built of memory functions from the lowest sensory input to the highest forms of insight.This concept has been explored in the area of intelligent control as the principle of increasing intelligence with decreasing precision (Saridis, 1979).

The transition from memory-less neural networks to systems that incorporate the concepts of short-term memory closely compares to the transition between memory-less and memory capable digital circuits mentioned earlier. There are two common forms of memory circuits, and while both techniques make use of memory, one system creates outputs that are solely dictated by the current state of the machine while the second approach allows the output of a given state to be altered immediately by the inputs to the system (as well as by the current state). Systems that do not force outputs to be synchronized with their internal state tend more often to generate unstable outputs. These momentary instabilities as the system output changes before and after the internal state can cause a race in the circuit. Instead of proceeding to the correct next state, the system jumps to a state determined by the intermediate outputs.

Clearly, the human model makes use of the synchronized technique for dealing with inputs (as described earlier the integrating functions of the sensory system synchronize inputs to the frequency of brain activity). This simple restriction insures the stability of the output at the cost of a more complex minimum circuit. Since the brain is made up of a large number of simple elements, the trade-off seems perfectly equatable. Thus the model of the human neural system should include memory elements to allow it to handle the concept of time, and a closer look should be given to the hierarchic structure of the thought process. This should provide a wide variety of solutions for many temporal problems, but it must be noted that other features of the human brain have been simplified here. Most notably, the fact that signals traversing the brain do not travel instantaneously and the fact that there are delays in the transmission of impulses from one neuron to the next could well account for some of the brains ability to compute time-based functions.

Once a network is capable of processing time dependent stimuli, the next logical step is to attempt to develop some form of look- ahead or precognition. The anticipation of events that may occur based on experience. For this transition, the system must be capable of applying a learned set of rules to multiple scenarios. Each application of the learned rule base would create additional possible futures. This process could be sequential (a recursive approach) or in parallel. From observation of human cognition, these two options are both implemented on a daily basis. For the more common occurrences, the parallel technique is subconsciously employed . The expected outcome of many situations is known and the necessary response is supplied automatically from associations. In other cases, a set of rules must be applied one step at a time in order to determine some future condition. The choice of process is usually determined by the level of familiarity one has with the task at hand. For instance, in a chess game a novice player will ponder each move and counter move where an expert player may simply attempt to visualize the possibilities. Each of these two methods has advantages: the parallel system would respond rapidly to a complex scenario while the sequential process would not. On the other hand, the sequential system has a greatly reduced learning time.

Either of these two techniques could provide interesting results when attempting to model time dependent response to dynamic systems. In order to implement a given approach the existing constraints of the neural network system must be considered. The back-propagation algorithm does not support the feedback loops that would be required to implement a recursive solution to a temporal problem. That is, there is no facility to route the solution or neural output at time **N** back into the same network in order to project a solution for time **N+1**. This limitation encourages experimentation towards the parallel approach. A gang of neural responses are grouped to accept data in a format which will be used to imply time passage. The outputs of previous layers will be passed into the input of an identical layer (representing the system at time **N+1**). Inputs to the system will be presented as a stream of sequential samples of time dependent data. Inputs for time **N** are given to the first layer. Inputs for time **N+1** are sent to the second layer and so on. In this manner a series of actions may be represented in the neural model as a large static problem where the flow of input signals is known for a brief span of time. Data presented in this manner may represent a history of data as well as the future inputs to the system. In the history mode, the weighting functions learned would represent the use of the previous states of the system in determining the proper course of action. While in the future data mode a system could be trained to expect a chain of events. Combination of the two techniques would provide a system that could respond correctly to sequences of stimuli.

III. Conclusions and Future Plans

This project has always recognized an eventual need to experiment on a large scale parallel architecture and a parallel machine was indicated in the original proposal as a specification task in the final year of the research. An opportunity occurred last summer when the Computer Engineering Department was offered an NCube parallel supercomputer at cost. NCube was interested in placing a machine in Florida and specifically at UCF. The MBR project elected to support roughly two-thirdsof the funding for the machine. The balance of funds was paid by the Office of Naval Research and the Computer Engineering Department. In September of 1990, Martin Marietta Orlando Aerospace agreed to fund the conversion of the simANNs system (see discussion above) to run as a parallel program on the NCube and be given a graphical user interface. This activity will have a strong impact on the MBR project as it will allow us to accelerate our original parallel programming plans by one year. Justifications for the use of parallel hardware beyond what we have discussed in our reports and presentations may be found in (Denning & Tichy, 1990)

At present we are continuing to expand and improve on the neural-based associative store and are assisting, when necessary, with the NCube port of our simANNs system. Our main focus now is on determining the best mechanism for temporal processing while maintaining our domain of launch processing.

References

Carpenter, G.A., Cohen, M.A. and Grossberg, S. (1987) Computing with Neural Networks, *Science*, **235**, 1226-1229.

Denning, P.J. and Tichy W.F. (1990) Highly Parallel Computation, *Science*, **250**, 1217-1222.

Dreyfus, H.L. (1979) *What Computers Can't Do*, New York: Harper and Row.

Dreyfus, H.L. and Dreyfus, S.E. (1988) Making a mind versus modeling the brain: AI back at a branchpoint, *Daedalus*, **117:1**,15-43.

Grossberg S. (1988) Nonlinear Neural Networks: Principles, Mechanisms, and Architectures, *Neural Networks*, **1**, 17-61.

Grossberg, S. and Schmajuk, N.A. (1989) Neural Dynamics of Adaptive Timing and Temporal Discrimination During Associative Learning, *Neural Networks*, **2**, 79-102.

Heileman, G.L., (1989) An Object-Oriented Approach to the Simulation of Artificial Neural Networks, Ph.D. Dissertation, University of Central Florida.

Hopfield, J.J. (1982) Neural networks and systems with emergent collective computational abilities, *Proc. Natl'l. Acad. Sci.*, **79**, 2554-2558.

Hopfield, J.J. and Tank, D.W. (1986) Computing with Neural Circuits: A Model, *Science*, **233**, 625-633.

Kanigel, R. (1987) Learning at the Sub-Neural Level, *Mosaic*, **18:3**, 16-25.

Kohonen T. (1988) An Introduction to Neural Computing, *Neural Networks*, **1**, 3-16.

Lenat, D.J., Borning, A., McDonald, D., Taylor, C. and Weyer, S. (1983) KNOESPHERE: Building Expert Systems with Encyclopedic Knowledge, **IJCAI**, 167-169.

Minsky, M. (1985) *The Society of Mind*, New York: Simon and Schuster.

Myler, H.R. (1991) A Neural Model for Memory-Based Reasoning in Advanced Launch Operations, *Journal of Neural Network Computing*, at press.

Saridis, G.N. (1979) Toward the realization of intelligent controls, *Proc. of the IEEE*, **67**.

Schiffren, R.M. and Atkinson, R.C.(1969) Storage and Retreival Processes in Long Term Memory, *Psychological Review*, **76**, 179-193.

Stanfill, C. and Waltz, D. (1986) Toward Memory-Based Reasoning, *Comm. of the ACM*, **29:12**, 1213-1228.

Takeda, M. and Goodman J.W. (1986) Neural Networks for Computation: Number Representations and programming complexity, *Applied Optics*, **25:18**, 3033-3046.

Waldrop, M.M. (1985) Machinations of Thought, *Science*, **85**, 37-52.

Appendix A Results of Multiple Training Runs of the Initial Topology

Back-Prop (learning rate 4)

number of patterns trained	1	Tolerance	.003 .009 .019
number of training passes	14	10	7
number of patterns trained	2	Tolerance	.003 .009 .019
number of training passes	224	18	10
number of patterns trained	3	Tolerance	.003 .009 .019
number of training passes	681	315	249
number of patterns trained	4	Tolerance	.003 .009 .019
number of training passes	760	396	324
number of patterns trained	5	Tolerance	.003 .009 .019
number of training passes	1900		385
number of patterns trained	6	Tolerance	.003 .009 .019
number of training passes	2490	852	684
number of patterns trained	7	Tolerance	.003 .009 .019
number of training passes	4788	1491	791
number of patterns trained	8	Tolerance	.003 .009 .019
number of training passes	4064	2024	1104

Delta-Bar-Delta (learning rate 4)

number of patterns trained	1	Tolerance	.003 .009 .019
number of training passes	14	9	7
number of patterns trained	2	Tolerance	.003 .009 .019
number of training passes	180	12	6
number of patterns trained	3	Tolerance	.003 .009 .019
number of training passes	534	336	279
number of patterns trained	4	Tolerance	.003 .009 .019
number of training passes	644	452	380
number of patterns trained	5	Tolerance	.003 .009 .019
number of training passes	1180		535
number of patterns trained	6	Tolerance	.003 .009 .019
number of training passes	1218	882	708
number of patterns trained	7	Tolerance	.003 .009 .019
number of training passes	2751	1596	1043
number of patterns trained	8	Tolerance	.003 .009 .019
number of training passes	2968		1240

Appendix B Subnetwork Training Results

SINGLE LAYER "DICTIONARY" SUBNETS:

PART1B.NET Scheduler sub-layer network [1,5,1,3]

Number of patterns trained	3
convergence	.050
number of training passes	1138

PART2B.NET Weather sub-layer network [1,5,8,6]

Number of patterns trained	6
convergence	.050
number of training passes	1410

PART3B.NET Diagnostics sub-layer network [1,5,2,3]

Number of patterns trained	3
convergence	.050
number of training passes	489

PARTB.NET Combines Parts 1B,2B,3B into one main net [2,15,11,1,9]

******* Simulation Result *******
Algorithm: Back-prop
Convergence Tolerance: 0.050
Learning Rates ---
Layer 1: 0.00
Layer 2: 4.00
Number of passes:
Layer 1 ---
Forward: 7047
Backward: 1994
Layer 2 ---
Forward: 7047
Backward: 1994
Reached solution: Yes

PARTC.NET Results in binary pattern, representing output [3,15,11,1,3]
******* Simulation Result *******
Algorithm: Back-prop
Convergence Tolerance: 0.050
Learning Rates ---
Layer 1: 0.00
Layer 2: 0.00
Layer 3: 4.00
Number of passes:
Layer 1 ---
Forward: 100008
Backward: 100008
Layer 2 ---
Forward: 100008
Backward: 100008
Layer 3 ---
Forward: 100008
Backward: 100008
Reached solution: No (note: did not converge)

PARTD.NET Binary representation of individual output [2,15,11,3,9]

******* Simulation Result *******
Algorithm: Back-prop
Convergence Tolerance: 0.050
Learning Rates ---
Layer 1: 0.00
Layer 2: 4.00
Number of passes:
Layer 1 ---
Forward: 10917
Backward: 3920
Layer 2 ---
Forward: 10917
Backward: 3920
Reached solution: Yes

PARTE.NET Float value representation of output pattern [3,15,11,3,5,9]

******* Simulation Result *******
Algorithm: Back-prop
Convergence Tolerance: 0.009
Learning Rates ---
Layer 1: 0.00
Layer 2: 0.00
Layer 3: 4.00
Number of passes:
Layer 1 ---
Forward: 27
Backward: 2
Layer 2 ---
Forward: 27
Backward: 2
Layer 3 ---
Forward: 27
Backward: 2
Reached solution: Yes

PARTF.NET ASCII 5 bit representation of result[3,15,11,3,5,9]

******* Simulation Result *******
Algorithm: Back-prop
Convergence Tolerance: 0.100
Learning Rates ---
Layer 1: 0.00
Layer 2: 0.00
Layer 3: 4.00
Number of passes:
Layer 1 ---
Forward: 72
Backward: 53
Layer 2 ---
Forward: 72
Backward: 53
Layer 3 ---
Forward: 72
Backward: 53
Reached solution: Yes